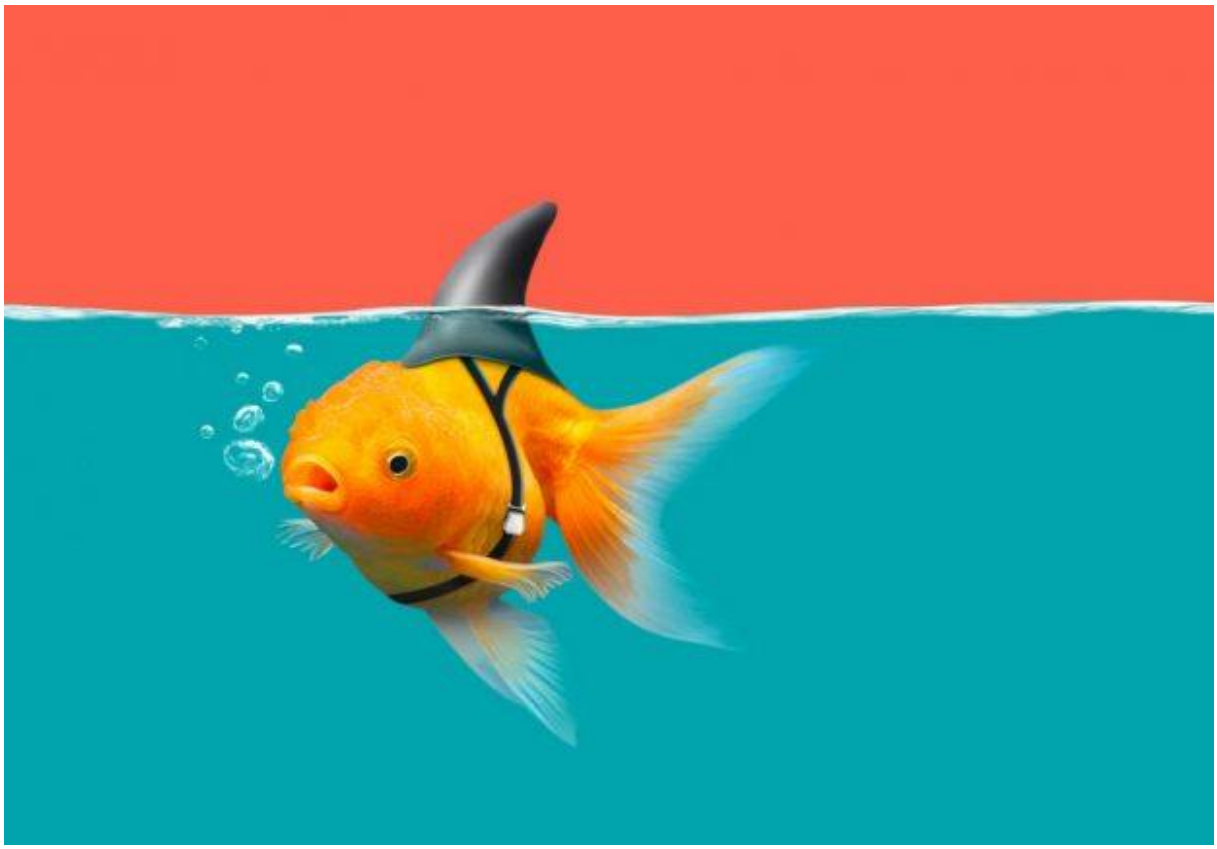
АНТИВИРУСНАЯ ЗАЩИТА БИЗНЕС-КЛАССА

ESET: новые схемы доставки бэкдора кибергруппы OceanLotus

03 апреля 2019 года

В посте расскажем, как кибергруппа OceanLotus (APT32 и APT-C-00) недавно использовала один из общедоступных эксплойтов к CVE-2017-11882, уязвимости повреждения памяти в Microsoft Office, и как вредоносное ПО группы обеспечивает персистентность в скомпрометированных системах, не оставляя следов. Дальше опишем, как с начала 2019 года группа использовала самораспаковывающиеся архивы для запуска кода.

OceanLotus специализируется на кибершпионаже, приоритетные цели – страны Юго-Восточной Азии. Атакующие подделывают документы, привлекающие внимание потенциальных жертв, чтобы убедить тех выполнить бэкдор, а также работают над развитием инструментария. Методы, используемые для создания приманок, варьируются в разных атаках – от файлов с «двойным расширением», самораспаковывающихся архивов, документов с макросами до известных эксплойтов.



Использование эксплойта в Microsoft Equation Editor

В середине 2018 года OceanLotus провела кампанию с использованием уязвимости CVE-2017-11882. Один из вредоносных документов кибергруппы проанализировали специалисты 360 Threat Intelligence Center (исследование на китайском), включив детальное описание эксплойта. В посте ниже – обзор подобного вредоносного документа.

Первый этап

Документ `FW Report on demonstration of former CNRP in Republic of Korea.doc` (SHA-1: `D1357B284C951470066AAA7A8228190B88A5C7C3`) аналогичен упомянутому в исследовании выше. Он интересен тем, что нацелен на пользователей, интересующихся камбоджийской политикой (CNRP – Партия национального спасения Камбоджи, распущенная в конце 2017 года). Несмотря на расширение .doc, документ имеет формат RTF (см. рисунок ниже), содержит мусорный код, а также искажен.

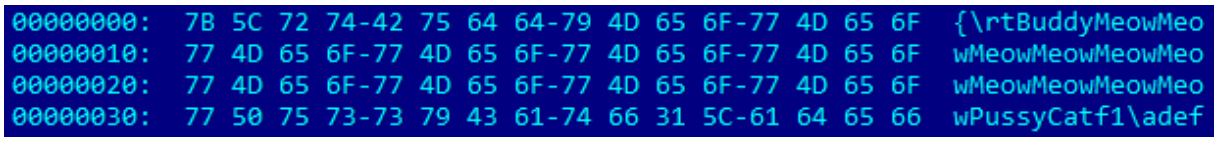


Рисунок 1. «Мусор» в RTF

Несмотря на наличие искаженных элементов, Word успешно открывает этот RTF-файл. Как видно из рисунка 2, здесь структура EQNOLEFILEHDR со смещением 0xC00, за которой следует заголовок MTEF, а затем запись MTEF (рисунок 3) для шрифта.

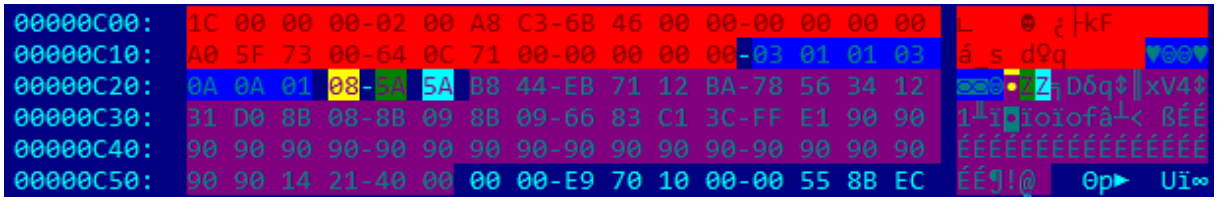


Рисунок 2. Значения записи FONT

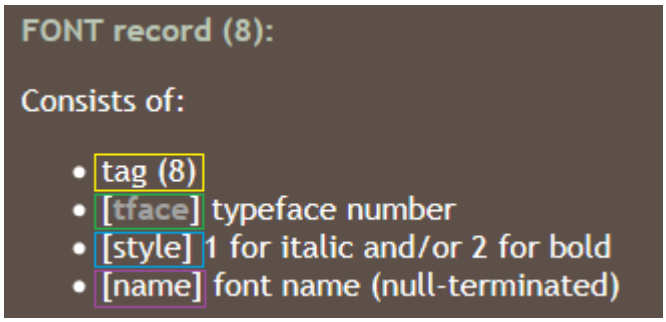


Рисунок 3. Формат записи FONT

Возможно переполнение в поле *name*, поскольку его размер не проверяется перед копированием. Слишком длинное имя запускает уязвимость. Как видно из содержимого RTF-файла (смещение 0xC26 на рисунке 2), буфер заполняется шелл-кодом, за которым следует фиктивная команда (`0x90`) и адрес возврата `0x402114`. Адрес является диалоговым элементом в `EQNEDT32.exe`, указывающим на инструкцию `RET`. Это приводит к тому, что EIP указывает на начало поля *name*, содержащего шелл-код.

```
seg000:00000C26
seg000:00000C26                                 public shellcode_start
seg000:00000C26                 shellcode_start proc near
seg000:00000C26 B8 44 EB 71 12                  mov     eax, 1271EB44h
seg000:00000C2B BA 78 56 34 12                  mov     edx, 12345678h
seg000:00000C30 31 D0                           xor     eax, edx        ; 0x45bd3c
seg000:00000C32 8B 08                           mov     ecx, [eax]
seg000:00000C34 8B 09                           mov     ecx, [ecx]
seg000:00000C36 8B 09                           mov     ecx, [ecx]
seg000:00000C38 66 83 C1 3C                     add     cx, 3Ch ; '<'
seg000:00000C3C FF E1                           jmp     ecx             ; jump to 0xc58
seg000:00000C3C                 shellcode_start endp
seg000:00000C3C
seg000:00000C3C                 ; ---------------------------------------------------------
seg000:00000C3E 90 90 90 90 90 90 90+nop_sled   db 14h dup(90h)
seg000:00000C52 14 21 40 00             ret_gadget      dd 402114h
```

Рисунок 4. Начало шелл-кода эксплойта

Адрес `0x45BD3C` хранит переменную, которая разыменовывается, пока не достигнет указателя на текущую загруженную структуру `MTEFData`. Здесь находится оставшаяся часть шелл-кода.

Назначение шелл-кода – выполнение второго фрагмента шелл-кода, встроенного в открытый документ. Сначала исходный шелл-код пытается найти дескриптор файла открытого документа, перебирая все дескрипторы системы (`NtQuerySystemInformation` с аргументом `SystemExtendedHandleInformation`) и проверяя, соответствуют ли *PID* дескриптора и *PID* процесса `WinWord` и был ли документ открыт с маской доступа – `0x12019F`.

Чтобы подтвердить обнаружение правильного дескриптора (а не дескриптора другого открытого документа), содержимое файла отображается с помощью функции `CreateFileMapping`, и шелл-код проверяет, соответствуют ли последние четыре байта документа "`yyyy`" (метод Egg Hunting). Как только будет обнаружено совпадение, документ копируется во временную папку (`GetTempPath`) как `ole.dll`. Затем читаются последние 12 байт документа.

```
00675F3F:  5D C2 04 00-50 FF 55 B0-8B 4D F4 E9-EC EB FF FF  ]┬♦ P U▒ï M⌠Θ∞δ
00675F4F:  AA BB CC DD-28 01 1D 00-79 79 79 79-             ¬╕H (Θ↔ yyyy
```

Рисунок 5. Маркеры конца документа

32-битное значение между маркерами `AABBCCDD` и `yyyy` – это смещение следующего шелл-кода. Он вызывается с помощью функции CreateThread. Извлечен тот же шелл-код, что использовался группой OceanLotus ранее. Скрипт эмуляции Python, который мы выпустили в марте 2018 года, все еще работает для дампа второго этапа.

Второй этап

Извлечение компонентов

Имена файлов и каталогов выбираются динамически. Код случайным образом выбирает имя исполняемого или DLL-файла в `C:\Windows\system32`. Затем он делает запрос к своим ресурсам и извлекает поле `FileDescription` для использования в качестве имени папки. Если это не работает, код случайным образом выбирает имя папки из каталогов `%ProgramFiles%` или `C:\Windows` (из GetWindowsDirectoryW). Он избегает использования имени, которое может конфликтовать с существующими файлами, и следит за тем, чтобы оно не содержало следующие слова: `windows`, `Microsoft`, `desktop`, `system`, `system32` или `syswow64`. Если каталог уже существует, к имени добавляется «NLS_{6 символов}».

Ресурс `0x102` анализируется и файлы сбрасываются в `%ProgramFiles%` или `%AppData%`, в папку, выбранную случайным образом. Время создания изменено, чтобы иметь те же значения, что и у `kernel32.dll`.

Например, вот папка и список файлов, созданных путем выбора исполняемого файла `C:\Windows\system32\TCPSVCS.exe` в качестве источника данных.

```
C:\Users\          \AppData\Roaming\TCPIP Services Application>dir
 Volume in drive C has no label.
 Volume Serial Number is

 Directory of C:\Users\          \AppData\Roaming\TCPIP Services Application

11/20/2010  10:24 PM    <DIR>          .
11/20/2010  10:24 PM    <DIR>          ..
11/20/2010  10:24 PM        11,715,584 Flash Video Extension.dll
11/20/2010  10:24 PM               345 Microsoft.VC80.CRT.manifest
11/20/2010  10:24 PM        11,715,584 MSVCP80.dll
11/20/2010  10:24 PM        11,715,584 MSVCR80.dll
11/20/2010  10:24 PM         1,636,558 TCPSVCS.db3
11/20/2010  10:24 PM            66,944 TCPSVCS.exe
               6 File(s)     36,850,599 bytes
               2 Dir(s)  45,739,352,064 bytes free
```

Рисунок 6. Извлечение различных компонентов

Структура ресурса `0x102` в дроппере достаточно сложна. В двух словах, он содержит:
— Имена файлов
— Размер и содержание файлов
— Формат сжатия (`COMPRESSION_FORMAT_LZNT1`, используемый функцией `RtlDecompressBuffer`)

Первый файл сбрасывается как `TCPSVCS.exe`, являющийся легитимным `AcroTranscoder.exe` (согласно `FileDescription`, SHA-1: `2896738693A8F36CC7AD83EF1FA46F82F32BE5A3`).

Возможно, вы заметили, что размер некоторых DLL-файлов превышает 11 Мб. Это связано с тем, что большой непрерывный буфер случайных данных размещается внутри исполняемого файла. Не исключено, что это способ избежать обнаружения некоторыми продуктами для безопасности.

Обеспечение персистентности

Ресурс `0x101` в дроппере содержит два 32-битных целых числа, которые определяют, каким образом следует обеспечить персистентность. Значение первого указывает, как вредоносная программа будет сохранять персистентность без прав администратора.

Значение первого числа	Механизм персистентности
0	Не обеспечивает персистентность
1	Запланированное задание от текущего пользователя
2	(HKLM|HKCU)\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
3	Создание файла ярлыка (с расширением .lnk) в подкаталоге Microsoft\Windows\Start Menu\Programs\Startup под одной из переменных среды: %ALLUSERSPROFILE%, %APPDATA% или %USERPROFILE%

Таблица 1. Механизм персистентности без прав администратора

Значение второго целого числа указывает, как вредоносная программа должна обеспечить персистентность, работая с правами администратора.

Значение второго числа	Механизм персистентности
1	Запланированное задание от администратора
2	Создание службы

Таблица 2. Механизм персистентности с правами администратора

Имя службы – это имя файла без расширения; отображаемое имя – имя папки, но если оно уже существует, к нему добавляется строка “`Revision 1`” (число увеличивается до тех пор, пока не будет найдено неиспользуемое имя). Операторы позаботились о том, чтобы персистентность через службу была устойчивой – в случае сбоя служба должна быть перезапущена через 1 секунду. Затем значению `WOW64` нового ключа реестра службы присваивается значение 4, что указывает на то, что это 32-разрядная служба.

Запланированное задание создается через несколько COM интерфейсов: `ITaskScheduler`, `ITask`, `ITaskTrigger`, `IPersistFile` и `ITaskScheduler`. По сути, вредоносная программа создает скрытую задачу, устанавливает информацию об учетной записи вместе с информацией о текущем пользователе или администратора и затем задает триггер.

Это ежедневное задание с продолжительностью 24 часа и интервалами между двумя выполнениями в 10 минут, что означает, что оно будет выполняться постоянно.

Вредоносный бит

В нашем примере исполняемый файл `TCPSVCS.exe` (`AcroTranscoder.exe`) является легитимным ПО, загружающим DLL, которые сбрасываются вместе с ним. В этом случае интерес представляет `Flash Video Extension.dll`.

Его функция `DLLMain` просто вызывает еще одну функцию. Присутствуют некоторые нечеткие предикаты:

```
result = 0;
if ( "0E8j2kP9zUe9VdsFEfg2H1BV3EZAbhSKZjI52IH2pvAGji18Bi7abkTQf0ebqLPbL3erPszpTar6BuA" )
{
  result = StrStrIA((LPCSTR)"0E8j2kP9zUe9VdsFEfg2H1BV3EZAbhSKZjI52IH2pvAGji18Bi7abkTQf0ebqLPbL3erPszpTar6BuA", "0");
  if ( result )
  {
    if ( result <= (LPSTR)"2kP9zUe9VdsFEfg2H1BV3EZAbhSKZjI52IH2pvAGji18Bi7abkTQf0ebqLPbL3erPszpTar6BuA" )
```

Рисунок 7. Нечеткие предикаты

После этих вводящих в заблуждение проверок код получает раздел `.text` файла `TCPSVCS.exe`, меняет его защиту на `PAGE_EXECUTE_READWRITE` и перезаписывает его, добавляя фиктивные инструкции:

```
nopsled:
                dec     ecx
                push    edi
                push    edi
                nop
                xchg    eax, ebx
                dec     ecx
                xchg    eax, ebx
                dec     ebx
                xchg    eax, ebx
                xchg    eax, ebx
                dec     eax
                dec     eax
                xchg    eax, ebx
                dec     edx
                xchg    eax, ebx
                push    ebp
                push    ecx
                inc     ebx
                inc     edx
                dec     eax
                nop
```

Рисунок 8. Последовательность инструкций

В конце к адресу функции `FLVCore::Uninitialize(void)`, экспортируемой `Flash Video Extension.dll`, добавляется инструкция `CALL`. Это означает, что после загрузки вредоносной DLL, когда среда выполнения вызывает `WinMain` в `TCPSVCS.exe`, указатель инструкции будет указывать на NOP, в результате вызывая `FLVCore::Uninitialize(void)`, следующий этап.

Функция просто создает мьютекс, начинающийся с `{181C8480-A975-411C-AB0A-630DB8B0A221}`, за которым следует текущее имя пользователя. Затем она читает сброшенный файл с расширением *.db3, который содержит позиционно-независимый код, и использует `CreateThread` для выполнения содержимого.

Содержимое файла *.db3 представляет собой шелл-код, который обычно использует группа OceanLotus. Мы вновь успешно распаковали его полезную нагрузку, используя скрипт эмулятора, который мы опубликовали на GitHub.

Скрипт извлекает финальный этап. Этот компонент является бэкдором, который мы уже проанализировали в предыдущем исследовании OceanLotus. Это можно определить по GUID `{A96B020F-0000-466F-A96D-A91BBF8EAC96}` бинарного файла. Конфигурация вредоносного ПО все еще зашифрована в PE ресурсе. Он имеет примерно ту же конфигурацию, но C&C-серверы отличаются от прежних:

- `andreagahuvrauvin[.]com`
- `byronorenstein[.]com`

- stienollmache[.]xyz

Группа OceanLotus снова демонстрирует сочетание разных техник, чтобы избегать обнаружения. Они вернулись с «доработанной» схемой процесса заражения. Выбирая случайные имена и заполняя исполняемые файлы случайными данными, они уменьшают число надежных IoC (на основе хешей и имен файлов). Более того, благодаря использованию сторонней загрузки DLL, атакующим нужно лишь удалить легитимный бинарник `AcroTranscoder`.

Самораспаковывающиеся архивы

После RTF-файлов группа перешла на самораспаковывающиеся (SFX) архивы с распространенными иконками документов, чтобы еще больше запутать пользователя. Об этом писали Threatbook (ссылка на китайском). После запуска сбрасываются самораспаковывающиеся RAR-файлы и исполняются DLL с расширением .ocx, финальная полезная нагрузка которых ранее была задокументирована `{A96B020F-0000-466F-A96D-A91BBF8EAC96}.dll`. С середины января 2019 года OceanLotus повторно используют эту технику, но со временем меняют некоторые конфигурации. В данном разделе мы расскажем о технике и изменениях.

Создание приманки

Документ `THICH-THONG-LAC-HANH-THAP-THIEN-VIET-NAM (1).EXE` (SHA-1: `AC10F5B1D5ECAB22B7B418D6E98FA18E32BBDEAB`) впервые найден в 2018 году. Этот SFX файл создан с умом – в описании (*Version Info*) сказано, что это изображение JPEG. Скрипт SFX выглядит следующим образом:

```
;The comment below contains SFX script commands

Setup=regsvr32 /s /i {9ec60ada-a200-4159-b310-8071892ed0c3}.ocx
Setup=regsvr32 /s /i {9ec60ada-a200-4159-b310-8071892ed0c3}.ocx
Setup="2018 thich thong lac.jpg"
TempMode
Overwrite=1
```

Рисунок 9. Команды SFX

Вредоносная программа сбрасывает `{9ec60ada-a200-4159-b310-8071892ed0c3}.ocx` (SHA-1: `EFAC23B0E6395B1178BCF7086F72344B24C04DCC`), а также картинку `2018 thich thong lac.jpg`.

Изображение-приманка выглядит следующим образом:

Рисунок 10. Изображение-приманка

Возможно, вы заметили, что первые две строки в скрипте SFX дважды вызывают OCX файл, но это не ошибка.

{9ec60ada-a200-4159-b310-8071892ed0c3}.ocx (ShLd.dll)

Поток управления OCX файла очень похож на другие компоненты OceanLotus – много последовательностей команд `JZ/JNZ` и `PUSH/RET`, чередующихся с мусорным кодом.

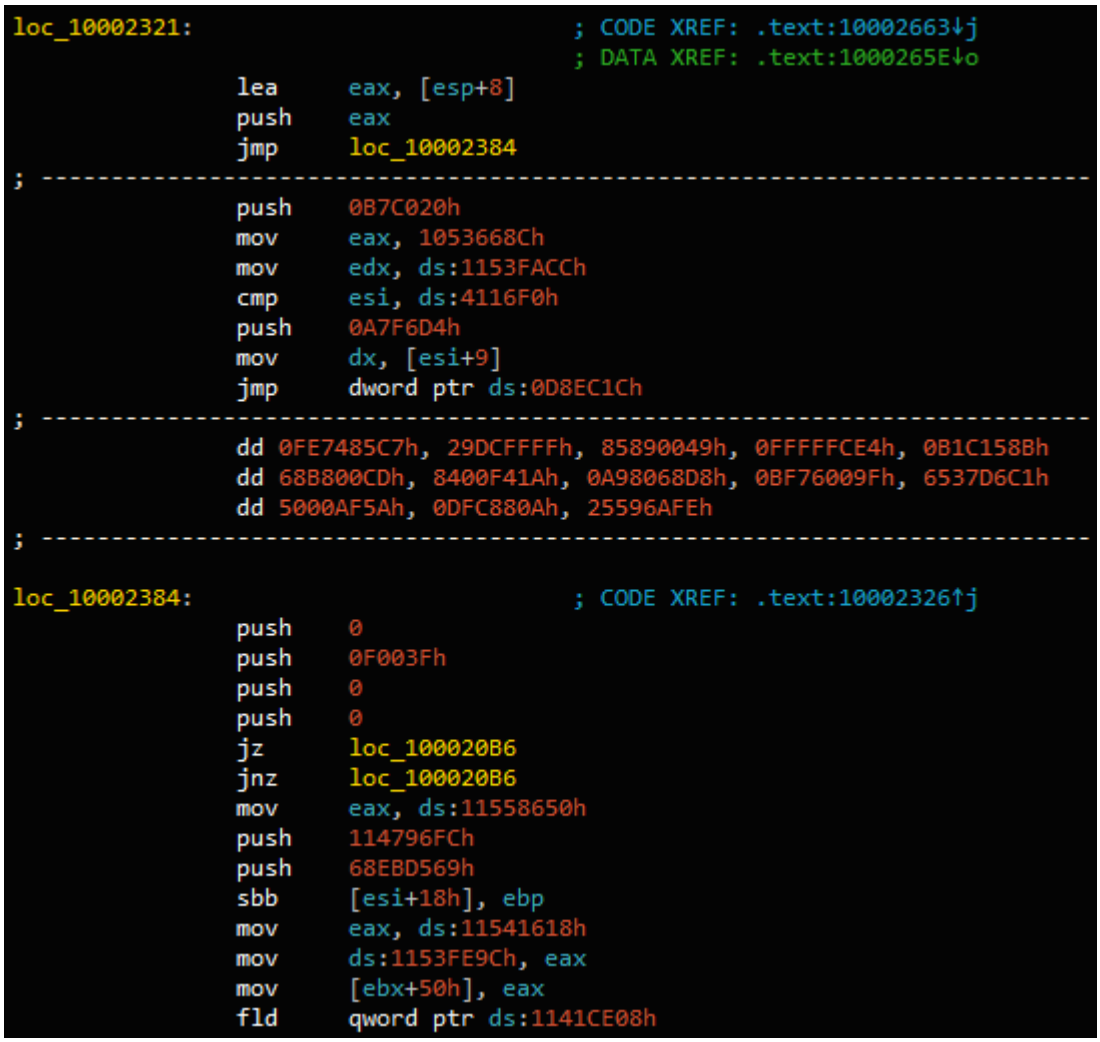


Рисунок 11. Обфусцированный код

После фильтрации мусорного кода экспорт `DllRegisterServer`, вызываемый `regsvr32.exe`, выглядит следующим образом:

```
LSTATUS sub_10001F70()
{
  LSTATUS result; // eax
  LSTATUS v1; // edi
  int DllBaseAddress; // esi
  BYTE Data[4]; // [esp+7962h] [ebp-10h]
  HKEY phkResult; // [esp+7966h] [ebp-Ch]
  DWORD cbData; // [esp+796Ah] [ebp-8h]
  DWORD Type; // [esp+796Eh] [ebp-4h]

  phkResult = 0;
  result = RegCreateKeyExW(
             HKEY_CURRENT_USER,
             L"SOFTWARE\\Classes\\CLSID\\{E08A0F4B-1F65-4D4D-9A09-BD4625B9C5A1}",
             0,
             0,
             0,
             KEY_ALL_ACCESS,
             0,
             &phkResult,
             0);
  if ( phkResult )
  {
    DllBaseAddress = f_GetDllBaseAddress();
    Type = 0;
    *(_DWORD *)Data = 0;
    cbData = 4;
    v1 = RegQueryValueExW(phkResult, L"Model", 0, &Type, Data, &cbData);
    Sleep(0x3E8u);
    if ( v1 || !*(_DWORD *)Data || cbData < 4 || Type != 4 )
    {
      *(_DWORD *)Data = 0x125211 - DllBaseAddress + f_Ret_10001DE0();
      RegSetValueExW(phkResult, L"Model", 0, 4u, Data, 4u);
      result = RegCloseKey(phkResult);
    }
    else
    {
      RegDeleteValueW(phkResult, L"Model");
      *(_DWORD *)Data = *(_DWORD *)Data + DllBaseAddress - 0x125211;
      (*(void (__stdcall **)(int))Data)(DllBaseAddress);
      result = RegCloseKey(phkResult);
    }
  }
  return result;
```

Рисунок 12. Основной код установщика

По сути, при первом вызове `DllRegisterServer` экспорт устанавливает значение реестра `HKCU\SOFTWARE\Classes\CLSID\{E08A0F4B-1F65-4D4D-9A09-BD4625B9C5A1}\Model` для зашифрованного оффсета в DLL (`0x10001DE0`).

Когда функция вызывается во второй раз, она читает то же значение и выполняется по этому адресу. Отсюда читается и выполняется ресурс и много действий в оперативной памяти.

Шелл-код – тот же загрузчик PE, используемый в прошлых кампаниях OceanLotus. Его можно эмулировать с помощью нашего скрипта. В итоге он сбрасывает `db293b825dcc419ba7dc2c49fa2757ee.dll`, загружает его в память и выполняет `DllEntry`.

DLL извлекает содержимое своего ресурса, расшифровывает (AES-256-CBC) и распаковывает (LZMA) его. Ресурс имеет специфический формат, который легко декомпилировать.

Рисунок 13. Структура конфигурации установщика (KaitaiStruct Visualizer)

Конфигурация задана в явном виде – в зависимости от уровня привилегий бинарные данные будут записываться в `%appdata%\Intel\logs\BackgroundUploadTask.cpl` или `%windir%\System32\BackgroundUploadTask.cpl` (или `SysWOW64` для 64-битных систем).

Далее персистентность обеспечивается путем создания задачи с именем `BackgroundUploadTask[junk].job`, где `[junk]` представляет собой набор байтов `0x9D` и `0xA0`.

Имя приложения задачи `%windir%\System32\control.exe`, а значение параметра – путь к выгруженному бинарному файлу. Скрытая задача запускается каждый день.

Конструктивно файл CPL представляет собой DLL с внутренним именем `ac8e06de0a6c4483af9837d96504127e.dll`, которая экспортирует функцию `CPlApplet`. Этот файл расшифровывает свой единственный ресурс `{A96B020F-0000-466F-A96D-A91BBF8EAC96}.dll`, затем загружает эту DLL и вызывает ее единственный экспорт `DllEntry`.

Файл конфигурации бэкдора

Конфигурация бэкдора зашифрована и встроена в его ресурсы. Структура файла конфигурации очень похожа на предыдущую.

```
[-] [root]
  [.] total_length = 474072
  [-] data_n
    [.] reg_part_len = 298
    [-] domain_encoding_str
      [.] str_len = 20
      [.] str_buf = "ghijklmnop"
    [-] first_reg
      [.] str_len = 122
      [.] str_buf = "SOFTWARE\\App\\AppX37cc7fdccd644b4f85f4b22d5a3f105a\\Application"
    [-] second_reg
      [.] str_len = 122
      [.] str_buf = "SOFTWARE\\App\\AppX37cc7fdccd644b4f85f4b22d5a3f105a\\DefaultIcon"
    [-] reg_value
      [.] str_len = 8
      [.] str_buf = "Data"
    [-] mutex_encoding_str
      [.] str_len = 6
      [.] str_buf = "abc"
    [.] domain_part_len = 104
    [-] domains_str
      [-] domain (3 = 0x3 entries)
        [-] 0
          [.] str_len = 30
          [.] str_buf = "ursulapapst.xyz"
        [-] 1
          [.] str_len = 30
          [.] str_buf = "sophiahoule.com"
        [-] 2
          [.] str_len = 32
          [.] str_buf = "karolinblair.com"
    [.] binaries_len = 473604
    [-] binaries
      [-] binaries (1 = 0x1 entries)
        [-] 0
          [.] bin_len = 473600
          [.] bin_data = 4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00 b8 00 00 00 00 00 00 00
    [.] backdoor_version_len = 4
    [.] backdoor_version = 23 45 80 10
    [.] unknown_len = 4
    [.] unknown = 2d 00 00 00
    [-] port_number
      [.] str_len = 10
      [.] str_buf = "14146"
    [.] unknown2_len = 8
    [.] unknown2 = ca 86 0f 38 08 5a 06 38
    [.] timeout_len = 8
    [.] timeout1 = 120000
    [.] timeout2 = 30000
```

Рисунок 14. Структура конфигурации бэкдора (KaitaiStruct Visualizer)

Несмотря на схожую структуру, значения многих полей были обновлены по сравнению с данными, приведенными в нашем старом отчете.

Первый элемент бинарного массива содержит DLL (`HttpProv.dll` MD5: `2559738D1BD4A999126F900C7357B759`), идентифицированную Tencent. Но поскольку имя экспорта было удалено из бинарного файла, хеши не совпадают.

Дополнительные исследования

Собирая образцы, мы обратили внимание на некоторые характеристики. Только что описанный образец появился примерно в июле 2018 года, а другие подобные ему – совсем недавно, в середине января – начале февраля 2019 года. В качестве вектора заражения использовался архив SFX, сбрасывающий легитимный документ-приманку и вредоносный файл OCX.

Несмотря на то, что OceanLotus используют поддельные временные метки, мы заметили, что

временные метки файлов SFX и OCX всегда одинаковые (`0x57B0C36A` (08/14/2016 @ 7:15pm UTC) и `0x498BE80F` (02/06/2009 @ 7:34am UTC) соответственно). Вероятно, это указывает на то, что у авторов есть некий «конструктор», который использует одни и те же шаблоны и просто меняет некоторые характеристики.

Среди документов, которые мы изучили с начала 2018 года, встречаются различные названия, указывающие на интересующие атакующих страны:

— *The New Contact Information Of Cambodia Media(New).xls.exe*
— *李建香(个人简历).exe (fake pdf document of a CV)*
— *feedback, Rally in USA from July 28-29, 2018.exe*

С момента обнаружения бэкдора `{A96B020F-0000-466F-A96D-A91BBF8EAC96}.dll` и публикации его анализа несколькими исследователями мы наблюдали некоторые изменения в данных конфигурации вредоносных программ.

Во-первых, авторы начали удалять имена из вспомогательных DLL DLLs (`DNSprov.dll` и две версии `HttpProv.dll`). Затем операторы прекратили упаковывать третью DLL (вторая версия `HttpProv.dll`), выбрав встраивание только одной.

Во-вторых, многие поля конфигурации бэкдора были изменены, вероятно, чтобы избежать обнаружения, поскольку многие IoCs стали доступны. В числе важных полей, измененных авторами, следующие:

- изменен раздел реестра AppX (см. IoCs)
- строка кодирования мьютекса («def», «abc», «ghi»)
- номер порта

Наконец, во всех новых проанализированных версиях появились новые C&C, перечисленные в разделе IoCs.

Выводы

OceanLotus продолжает развиваться. Кибергруппа сфокусирована на доработке и расширении инструментария и приманок. Авторы маскируют вредоносные полезные нагрузки с помощью привлекающих внимание документов, тема которых актуальна для предполагаемых жертв. Они разрабатывают новые схемы, а также используют общедоступные инструменты, например, эксплойт Equation Editor. Более того, они совершенствуют инструменты, чтобы уменьшить количество артефактов, остающихся на машинах жертв, тем самым снижая шанс на обнаружение антивирусным ПО.

Индикаторы компрометации

Индикаторы компрометации, а также атрибуты MITRE ATT&CK доступны на Welivesecurity и на GitHub.